Design of PeerSum: a Summary Service for P2P Applications

Rabab Hayek, Guillaume Raschia, Patrick Valduriez, Noureddine Mouaddib

Design of PeerSum: a Summary Service for P2P Applications

Rabab Hayek†, Guillaume Raschia†, Patrick Valduriez‡, and Noureddine Mouaddib†

Atlas team, INRIA and LINA, University of Nantes, France
†FirstName.LastName@univ-nantes.fr, ‡Patrick.Valduriez@inria.fr

Abstract. Sharing huge databases in distributed systems is inherently difficult. As the amount of stored data increases, data localization techniques become no longer sufficient. A more efficient approach is to rely on compact database summaries rather than raw database records, whose access is costly in large distributed systems. In this paper, we propose PeerSum, a new service for managing summaries over shared data in large P2P and Grid applications. Our summaries are synthetic, multidimensional views with two main virtues. First, they can be directly queried and used to approximately answer a query without exploring the original data. Second, as semantic indexes, they support locating relevant nodes based on data content. Our main contribution is to define a summary model for P2P systems, and the algorithms for summary management. Our performance evaluation shows that the cost of query routing is minimized, while incurring a low cost of summary maintenance.

1 Introduction

Research on distributed systems is focusing on supporting advanced applications which must deal with semantically rich data (e.g. XML documents, relational tables, etc.), using a high-level SQL-like query language. As a potential example of applications, consider the cooperation of scientists who are willing to share their private data for the duration of a given experiment. Such cooperation may be efficiently supported by improving the data localization and data description techniques.

Initially developed for moderate-sized scientific applications, Grid technology is now evolving to provide database sharing services, in large virtual organizations. In [9], a service-based architecture for database access (OGSA-DAI) has been defined over the Grid. OGSA-DAI extends the distributed database architecture [13] to provide distribution transparency using Web services. However, it relies on some centralized schema and directory management, which is not an adequate solution for supporting highly dynamic organizations, with a large number of autonomous members.

Peer-to-Peer (P2P) techniques that focus on scaling up, dynamicity, autonomy and decentralized control can be very useful to Grid data management. The complementary nature of the strengths and weaknesses of the two technologies suggests that the interests of the two communities are likely to grow closer over time [6]. For instance, P-Grid [1] and Organic Grid [3] develop self-organizing and scalable services on top of P2P systems.

In unstructured P2P systems, query routing relies on flooding mechanisms which suffer from high query execution cost and poor recall. To improve performance, several techniques have been proposed to locate data relevant to a user query. These techniques can be grouped in three classes: data indexing, mediation and content-based clustering. Data indexing maintains the location (e.g. [18], [15]) or the direction (e.g. [4]) to nodes storing relevant data. However, efficient data indexes must be small, distributed and refer to data based on their content, without compromising peer autonomy or mandating a specific network structure. Mediation consists in exploiting structural information on data schemas to guide query propagation. For instance, in Piazza [19], a query is propagated along pre-existing pairwise mappings between peer schemas. However, many limitations prevent these techniques from scaling up. Content-based clustering consists in organizing the network such that "similar" peers, e.g. peers answering similar queries, are grouped together ([12], [5]). Similarity between peers may be computed using techniques of the two preceding classes (e.g. similarity between indexes [11]).

With the ever increasing amount of information stored into databases, data localization techniques are no longer sufficient to support P2P data sharing. Today's Decision-Support and collaborative applications are typically exploratory. Thus, a user may prefer a fast, approximate answer to a long, exact answer. In other words, reasoning on compact data descriptions rather than raw database records, whose access is costly in large P2P systems, may be much more efficient. For instance, a doctor asking queries like "*young* and *fat* patients diagnosed with disease X" may prefer descriptions of result tuples to rapidly make a decision based on similar situations, treated by other doctors.

In this paper, we propose PeerSum, a new service for managing summaries over shared data in P2P systems. Our summaries are synthetic, multidimensional views with two main virtues. First, they provide an intelligible representation of the underlying data such that an approximate query can be processed entirely in their domain; that is, inputs and outputs are summaries. Second, as indexing structures, they support locating relevant nodes based on their data descriptions. PeerSum is done in the context of APPA, a network-independent P2P data management system [2].

This paper makes the following contributions. First, we define a summary model which deals with the distributed and autonomous nature of P2P systems. Second, we propose efficient algorithms for summary management. We validated our algorithmic solutions through simulation, using the BRITE topology generator and SimJava. The performance results show that the cost of query routing is minimized, while incurring a low cost of summary maintenance.

The rest of this paper is organized as follows. Section 2 describes PeerSum's summary model. Section 3 describes PeerSum's summary management with its algorithms. Section 4 discusses query processing with PeerSum. Section 5 gives a performance evaluation with a cost model and a simulation model. Section 6 compares our solution with related work. Section 7 concludes.

2 PeerSum summary model

In this section, we first present our summary model architecture and the principle of summary construction in P2P systems. Second, we discuss the scalability issues of

the summarization process that is integrated to a peer DataBase Management System (DBMS), to allow generating summaries of a relational database. Then, we formally define the notion of data summary in a P2P network.

2.1 Model architecture

Our ultimate goal is to build a complete summary that describes the content of all shared data sources. However, such a summary is ideal in the context of P2P networks, because of their autonomous and dynamic nature. It is difficult to build and to keep this summary consistent relative to the current data instances it describes. In our approach, we

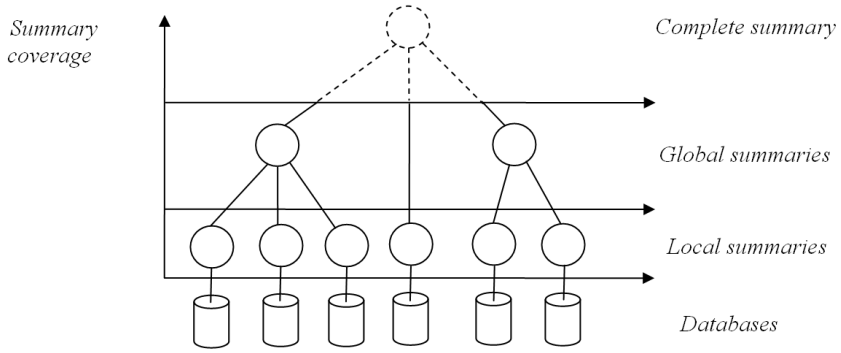

Fig. 1. Summary Model Architecture

adopt an incremental mechanism for summary construction, and define the notion of "*summary coverage*" as follows.

Definition 1. Summary coverage. *The* coverage *of a summary S in a network of size N is the fraction of the peers that own data described by the summary S.*

The coverage of a summary quantifies its convergence to the complete summary which is obviously characterized by a coverage = 1.

The architecture of our summary model is presented in Figure 1. Each peer generates the Local Summary (LS) of its database, which is characterized by the lowest-coverage level. Then, it cooperates with other peers through exchanging and merging summaries, in order to build a Global Summary (GS). The last one is characterized by a continuous evolution in term of coverage. In fact, the cooperation between two sets of peers, each having constructed a global summary, will result in a higher-coverage one. That is, in a large P2P system, one could see the global summary as an intermediate node in a global hierarchy where the virtual root is the ideal complete summary.

In this work, we propose fully distributed algorithms for global summary construction and maintenance. However, we will first give a brief description of the summarization process that generates summaries of relational databases with interesting features, making it scalable in a distributed environment.

2.2 Summarization process: Scalability issues

A summarization process is integrated to each peer's DBMS to allow constructing the local summary level of Figure 1. Our approach is based on SAINTETIQ [14], an on-line linguistic approach for summarizing databases. The system is organized into two separate web services. The *translation service* corresponds to the pre-processing step that prepares data for summarization while the *summarization service* produces a set of summaries arranged in a hierarchy. A unique feature of the summary system is its use of *Background Knowledge* (BK), a priori built on each attribute. It supports the translation of descriptions of database tuples into a user-defined vocabulary. Descriptors used for summary content representation are defined as linguistic variables [21] on the attribute domain. For example, Figure 2 shows a user-defined vocabulary on the attribute `age`. A detailed description of the SAINTETIQ process is available in [14] and [16]. Concerning our work, we are interested in the scalability of the summarization process in a distributed environment.

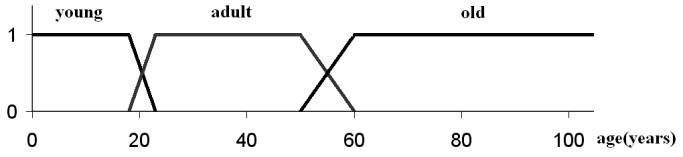

Fig. 2. Fuzzy Linguistic Partition on **`age`**

Memory consumption and time complexity are the two main factors that need to be taken care off in order to guaranty the capacity of the summary system to handle massive datasets. First, the process time complexity is in $O(n)$, where *n* is the number of tuples to incorporate into a hierarchy of summaries. Besides, an important feature is that in the summary algorithm raw data have to be parsed only once and it is performed with a low time cost. Second, the system requires low memory consumption for performing the summary construction algorithm as well as for storing the produced summaries. Moreover, a cache manager is in charge of summary caching in memory and it can be bounded to a given memory requirement. On the other hand, the parallelization of the summary system is a key feature to ensure a smooth scalability. The implementation of the system is based on the Message-Oriented Programming paradigm. Each sub-system is autonomous and collaborates with the others through disconnected asynchronous method invocations. It is among the least demanding approaches in terms of availability and centralization. The autonomy of summary components allows for a distributed computing of the process.

2.3 Summary representation

A summary z is a pair (I_z, R_z) where I_z is the intentional content of the summary and R_z is its extent, that is the group of database tuples described by I_z. The intent I_z

provides a short description of z in terms of linguistic labels defined in the Background Knowledge (BK) and used in the pre-processing step.

For our purpose, we consider a summary as an indexing structure over distributed data in a P2P system. Thus, we added a third dimension to the definition of a summary z: a *peer-extent* P_z, which provides the set of peers having data described by z.

Definition 2. Peer-extent. *Let z be a node in a given hierarchy of summaries S, and P the set of all peers who participated to the construction of S. The* peer-extent P_z *of the summary z is the subset of peers owning, at least, one record of its extent R_z: $P_z = \{p \in P \mid R_z \cap R_p \neq \emptyset\}$, where R_p is the view over the database of node p, used to build summaries.*

Due to the above definition, we extend the notion of *data-oriented* summary in a given database, to a *source-oriented* summary in a given P2P network. In other words, our summary can be used as a database index (e.g. referring to relevant tuples), as well as a semantic index in a distributed database system (e.g. referring to relevant nodes).

A summary is an edge in the tree structure finally produced by the summarization service. The summary hierarchy S will be characterized by its *Coverage* in the P2P system; that is, the fraction of nodes (data sources) covered by S (see Definition1). Relative to the hierarchy S, we call *Partner Peer* a peer whose data is described by at least a summary node of S.

Definition 3. Partner peers. *The set of* Partner peers P_S *of a summary hierarchy S is the union of peer-extents of all the summary nodes: $P_S = \{\cup_{z \in S} P_z\}$.*

By now and for convenient purpose only, we designate by "summary" a hierarchy of summaries maintained in a P2P system, unless otherwise specified.

3 Summary management in PeerSum

We present PeerSum, a summary management service for P2P systems. First, we study the integration of PeerSum in an existing P2P architecture. Here we work in the context of APPA (Atlas Peer to Peer Architecture) [2]. Then, we propose algorithms for PeerSum's summary management. APPA has a network-independent architecture so it can be implemented over different types of P2P networks. APPA provides three layers of services: P2P network, basic services and advanced services. PeerSum is integrated at the advanced layer and defined based on the underlying services. Due to space limitations, we will only mention the services required for PeerSum definition. According to Section 2.1, PeerSum must address the following requirements:

- Peers construct individually their local summaries,
- Peers cooperate for exchanging and merging summaries into a global summary,
- Peers share a common storage in which the global summary is maintained.

The first point is addressed by integrating the summarization process, previously defined, to each peer's DBMS. Second, the peer linking and peer communication services of the APPA's P2P network layer allow peers to communicate and exchange messages

(through service calls), while cooperating for a global summary construction. However, two problems arise from the heterogeneous nature of peers in a P2P system. First, peers may have different processing and storage capabilities. Therefore, a main function of PeerSum is to ensure a distributed operation for summary merging. A partner peer that requires merging two summaries, calls the service which then delegates the right peers to perform merging calculations, using load balancing and distributed computing techniques. This function can be implemented since the summarization process, at each peer, can be distributed and parallelized, as discussed in Section 2.2.

Second, peers exchange summaries that are produced using local Background Knowledges (BKs). Thus, they may be represented in different vocabularies, making difficult their shared exploitation. In this work, we assume that the participants to a collaborative database application agree on a *Common Background Knowledge* (CBK) that will be used locally by each summarization process. An example of such a CBK is the Systematized Nomenclature of Medicine Clinical Terms (SNOMED CT) [10], which is a comprehensive clinical terminology covering diseases, clinical findings, and procedures.

On the other hand, several works have addressed the problem of semantic heterogeneity in advanced P2P applications (e.g. [19], [2]). Since our summaries are data structures that respect the original data schemas [16], we can assume that the techniques they proposed for a decentralized schema management can be also used to overcome the heterogeneity of summary representations, in the context of different BKs.

Finally, the P2P data management (P2PDM) service of the basic layer and the Key-based Storage and Retrieval (KSR) service of the P2P network layer, work together to provide a common storage in which a global summary is maintained. This common storage increases the probability that "*P2P data*" (e.g. metadata, indexes, summaries) produced and used by advanced services are available even if peers that have produced them are disconnected. P2PDM and KSR manage data based on keys. A key is a data identifier which determines which peer should store the data in the system, e.g. through hashing over all peers in DHT networks or using super-peers for storage and retrieval in super-peer networks. All data operations on the common storage are key-based, i.e. they require a key as parameter.

In the following, we will describe our algorithms for summary construction and maintenance. First, we work in a static context where all the participants remain connected. Then, we address the dynamicity of peers and propose appropriate solutions.

3.1 Summary construction

Starting up with a local summary level (see Figure 1), we present the algorithm for peer cooperation that allows constructing a global summary GS. We assume that each global summary is associated with a *Cooperation List* (CL) that provides information about its partner peers. An element of the cooperation list is composed of two fields. A partner peer identifier *PeerID*, and a *2*-bit freshness value v that provides information about the freshness of the descriptions as well as the availability of the corresponding database.

- value 0 (initial value): the descriptions are fresh relative to the original data,
- value 1: the descriptions need to be refreshed,

– value 2: the original data are not available. This value will be used while addressing peer volatility in Section 3.3.

Both the global summary and its cooperation list are considered as "summary data" and are maintained in the common storage, using the P2PDM and KSR services.

Cooperation request The algorithm starts at an *initiator peer* P_{init} who sends a cooperation request message to its neighbors, to participate to a global summary construction. This message contains P_{init}'s identifier and a given value of TTL (Time-To-Live). One may think that a large value of TTL allows to obtain directly a high-coverage summary. However, due to the autonomous nature of P2P systems, P_{init} may keep waiting for a very long time without having constructed that global summary. Therefore, we choose to limit the value to TTL and adopt an incremental construction mechanism, as discussed in Section 2.1.

Cooperation response A peer *p* who receives the message, performs the following steps. First, if the request has already been received, it discards the message. Else, it saves the address of the sender as its parent. Then, its decrements TTL by one. If the value of TTL remains positive, it sends the message to its neighbors (except the *parent*) with the new TTL value. After propagating the message, *p* must wait to receive the responses of its neighbors. However, since some of the neighbors may leave the system and never response, the waiting time must be limited. We compute *p*'s waiting time using a cost function based on TTL, and network dependent parameters.

A cooperation response of a peer p has the following structure: $Coop_Resp = \langle$CS, PeerIDs, GSKeys$\rangle$. CS is the current summary obtained at p, $PeerIDs$ is the list of identifiers of peers that have responded to p, and $GSKeys$ is the list of keys of global summaries. If *p* is a partner peer, that is, *p* has already participated to an existing global summary, its $Coop\_Resp$ will include the key of the global summary it knows, as well as the peer identifiers contained in the corresponding *CL*, i.e. $Coop\_Resp = \langle \emptyset$, extractPeerIDs(CL), $\{$GSKey$\}\rangle$. In that case, *p* locates at the boundary of two knowledge scopes of two different summaries. Hence, it allows merging them into a higher-coverage one (i.e. incremental construction). Otherwise, its response will include its local summary and its identifier, i.e. $Coop_Resp = \langle$p.LS, $\{$p.ID$\}$, $\emptyset\rangle$.

Summary data storage In the waiting phase, when a child's $Coop\_Resp$ arrives, a parent peer *p* merges it with its own response by making the union of $PeerIDs$ and $GSKeys$ lists, and merging the current summaries. Once the time expires, *p* sends the result to its parent. But, if *p* is the initiator peer P_{init}, it will store the new summary data, i.e. the new global summary GS and its cooperation list CL, using the KSR service: *GSKey := KSR_insert(CS, CL)*. CL contains each peer identifier obtained in the final $PeerIDs$ list, associated with a freshness value v equal to zero. At the end, P_{init} sends the new key (*GSKey*) to all participant peers, which become GS's *partner peers*.

3.2 Summary maintenance

A crucial issue for any indexing structure is to maintain the index, relative to the current data instances, without incurring high costs. For a local summary, it has been demonstrated that the summarization process guarantees an incremental maintenance, using a *push* mode for exchanging data with the DBMS, while performing with a low complexity. In this section, we propose a strategy for maintaining a global summary based on both *push* and *pull* techniques, in order to minimize the number of messages exchanged in the system. The appropriate algorithm is divided into two phases: Data modification and summary reconciliation.

Push: Data modification Let GS be a global summary and P_{GS} the set of partner peers. Each partner is responsible for refreshing its own element in the GS's cooperation list. A partner peer p observes the modification rate issued on its local summary LS. When LS is considered as enough modified, p sets its freshness value v to 1, through a *push message*. This value indicates that the local summary version being merged while building GS does not correspond any more to the current instance of the database.

An important feature is that the frequency of push messages depends on modifications issued on local summaries, rather than on the underlying databases. It has been demonstrated in [16] that, after a given process time, a summary becomes very stable. As more tuples are processed, the need to adapt the hierarchy decreases. A summary modification can be determined by observing the appearance/disappearance of descriptors in summary intentions.

Pull: Service-Initiated reconciliation The summary service, in its turn, observes the fraction of old descriptions (i.e. number of ones) in the cooperation list. Whenever this fraction exceeds a threshold value, the global summary GS must be refreshed. In that case, the service pulls all the partner peers to merge their current local summaries into the new version of GS, which will be then under reconstruction. The algorithm is described as follows.

A reconciliation message that contains a new summary *NewGS* (initially empty), is propagated from a partner to another. When a partner *p* receives this message, it first merges NewGS with its local summary. Then, it sends the message to another partner (chosen from the cooperation list CL). If *p* is the last visited peer, it updates the GS's *summary data*, using the KSR service. All the freshness values in CL are reset to zero. This strategy guarantees a high availability of the summary data, since only one KSR_Update operation is performed by the last partner.

3.3 Peer dynamicity

In large P2P systems, a peer connects mainly to download some data and may leave the system without any constraint. Therefore, the shared data can be submitted to a low modification rate, while the rate of node arrival/departure is very important. We propose now solutions for that peer dynamicity.

Peer arrival When a new peer *p* joins the system, it contacts some existing peers to determine the set of its neighbors. If one of those neighbors is a partner peer, *p* becomes a new partner: a new element is added to the cooperation list with a freshness value v equal to one. Recall that the value 1 indicates the need of pulling the peer to get new data descriptions. Furthermore, if *p* is a neighbor of two partners of two different summaries, it allows merging them in a higher-coverage one (Section 3.1).

Peer departure When a partner peer *p* decides to leave the system, it first sets its freshness value v to two in the cooperation list, through a push message. This value reminds the participation of the disconnected peer p to the corresponding global summary, but also indicates the unavailability of the original data. There are two alternatives to deal with such a freshness value. First, we can keep the data descriptions and use it, when a query is approximately answered using the global summary. A second alternative consists in considering the data descriptions as expired, since the original data are not accessible. Thus, a partner departure will accelerate the summary reconciliation initiating. In the rest of this paper, we adopt the second alternative and consider only a *1*-bit freshness value v: a value 0 to indicate the freshness of data descriptions, and a value 1 to indicate either their expiration or their unavailability. However, if *p* failed, it could not notify its partners by its departure. In that case, its data descriptions will remain in the global summary until we execute a new summary reconciliation. The reconciliation algorithm does not require the participation of a disconnected peer. The global summary GS is reconstructed, and descriptions of unavailable data will be then omitted.

4 Query processing

Now we discuss how a query Q, posed at a peer p, is processed. Our approach consists in querying at first the available summary . This allows an efficient peer localization since we exploit data descriptions rather than structural information on data schemas, in order to propagate the query. Besides, when an exact answer is not required, summaries can directly provide approximate answers without accessing original database records. Query processing proceeds in two phases: 1) query extension and 2) query evaluation.

4.1 Query extension

First, the query Q must be extended to a flexible query Q^* in order to be handled by a summary querying process. For instance, consider the following selection query Q[1]:

Select BMI From Patient Where age $\prec 30$ *And disease = "Malaria"*

This phase consists in replacing the original value of each selection predicate by the corresponding descriptors defined in the Background Knowledge (*BK*). According to the fuzzy partition of Figure 2, the above query is transformed to Q^*:

Select BMI From Patient Where age In $\{young, adult\}$ *And disease ="Malaria"*

[1] Body Mass Index (BMI) is the patient's body weight divided by the square of the height.

Let QS (resp.QS^*) be the *Query Scope* of query Q (resp.Q^*), that is; the set of peers that should be visited to answer the query. Obviously, the query extension phase may induce false positives in query results. To illustrate, a patient having 35 years old will be returned as an answer to the query Q^*, while the selection predicate on the attribute `age` of the original query Q is not satisfied. However, false negatives can not occur which is expressed by the following inclusion: $QS \subseteq QS^*$.

In the rest of this paper, we suppose that a user query is directly formulated using descriptors defined in the BK (i.e. $Q = Q^*$). As we discussed in the introduction of this work, a doctor that participates to a given medical collaboration, may ask query Q like "the BMI of *young* and *adult* patients diagnosed with *malaria*".Thus, we eliminate eventual false positives that result from query extension.

4.2 Query evaluation

This phase deals with matching a set of summaries organized in a hierarchy S, against the query Q. The query is transformed into a logical proposition P used to qualify the link between a summary node and the query. P is under a conjunctive form in which all descriptors appears as literals. In consequence, each set of descriptors yields on corresponding clause. For instance, the above query Q is transformed to $P =$ *(young OU adult) ET (malaria)*. A valuation function has been defined to valuate the proposition P in the context of a summary node z. Then, a selection algorithm performs a fast exploration of the hierarchy and returns the set Z_Q of most precise summaries that satisfy the query. For more details see [20]. Once Z_Q determined, the query evaluation process is able to achieve two distinct tasks depending on the user/application requirements: 1) Peer localization to return the original result records and 2) Summary answering to return approximate answers.

Peer localization Since the extended definition of a summary node z provides a peer-extent, i.e. the set of peers P_z having data described by its intent (see Definition 2), we can define the set P_Q of relevant peers for the query Q as follows: $P_Q = \{\cup_{z \in Z_Q} P_z\}$.

The query Q is directly propagated to these relevant peers. Thus, a distinctive feature of our approach is that the number of hops the queries makes to find the matching nodes is "ideally" reduced to one, and consequently, excessive delays are avoided. However, the efficiency of this query routing depends on the completeness and the freshness of summaries, since stale answers may occur in query results. We define a *False Positive* as the case in which a peer p belongs to P_Q and there is actually no data in the p source that satisfies Q (i.e. $p \notin QS$). A *False Negative* is the reverse case in which a p does not belong to P_Q, whereas there exists at least one tuple in the p data source that satisfies Q (i.e. $p \in QS$).

Summary answering Another distinctive feature is that a query can be processed entirely in the summary domain. An approximate answer can be provided from summary descriptions, without having to access original, distributed database records. The selected summaries Z_Q are aggregated according to their interpretation of proposition P: summaries that have the same required characteristics on all predicates (i.e. *age* and

disease) form a class. The aggregation in a given class is a union of descriptors: for each attribute of the selection list (i.e. *BMI*), the querying process supplies a set of descriptors which characterize summaries that respond to the query through the same interpretation [20]. For example, for the class $\{$*young*, *malaria*$\}$, we can obtain an output set *BMI* = $\{$*underweight*, *normal*$\}$.

5 Performance evaluation

In this section, we devise a simple model of the summary management cost in PeerSum. Then, we evaluate and analyze our model with a simulation.

5.1 Cost model

A critical issue in summary management is to trade off the summary updating cost against the benefits obtained for queries.

Summary update cost Here, our first undertaking is to optimize the update cost while taking into account *query accuracy*. In the next section, we discuss query accuracy which is measured in terms of the percentage of false positives and false negatives in query results. The cost of updating summaries is divided into: usage of peer resources, i.e. time cost and storage cost, and the traffic overhead generated in the network.

Time cost: A unique feature of SAINTETIQ is that the changes in the database are reflected through an incremental maintenance of the summary hierarchy. The time complexity of the summarization process is in $O(n)$ where n is the number of tuples to be incorporated in that hierarchy [16]. For a global summary, we are concerned with the complexity of merging summaries. Recently, a new MERGING method has been proposed, based on the SAINTETIQ engine. This method consists in incorporating the leaves of a given summary hierarchy S_1 into an another S_2, using the same algorithm described by the SAINTETIQ summarization service. It has been proved that the complexity C_{M12} of the MERGING(S_1, S_2) process is constant w.r.t the number of tuples.

Storage cost: We denote by k the average size of a summary node. In the average-case assumption, there are $\sum_{i=0}^{d} B^i = (B^{d+1} - 1)/(B - 1)$ nodes in a B-arity tree with d, the average depth of the hierarchy. Thus the average space requirement is given by: $C_m = k.(B^{d+1} - 1)/(B - 1)$. Based on real test, $k = 512$ bytes gives a rough estimation of the space required for each summary node. An important issue is that the size of the hierarchy is quite related to its stabilization (i.e. B and d). As more tuples are processed, the need to adapt the hierarchy decreases and incorporating a new tuple may consist only in sorting a tree. Hence, the structure of the hierarchy remains stable and no additional space is required.

According to the above discussion, the usage of peer resources is optimized by the summarization process itself. Thus, we restrict now our focus to the traffic overhead generated in the P2P network.

Network traffic: Recall that there are two types of exchanged messages: *push* and *reconciliation*. Let local summaries have an average lifetime of L seconds in a given

global summary. Once L expired, the node sends a (push) message to update its freshness value v in the cooperation list CL. The reconciliation algorithm is then initiated whenever the following condition is satisfied: $\sum_{v \in CL} v/|CL| \geq \alpha$ where α is a threshold that represents the ratio of old descriptions tolerated in the global summary. During reconciliation, only one message is propagated among all partner peers until the new global summary version is inserted in the common storage. Let F_{rec} be the reconciliation frequency. The update cost is: $C_{up} = 1/L + F_{rec}$ messages per node per second. In this expression, $1/L$ represents the number of push messages which depends either on the modification rate issued on local summaries or the connection/disconnection rate of peers in the system. Higher is the rate, lower is the lifetime L, and thus a large number of push messages are entailed in the system. F_{rec} represents the number of reconciliation messages which depends on the value of α. This threshold is our system parameter that provides a trade-off between the cost of summary updating and query accuracy. If α is large, the update cost is low since a low frequency of reconciliation is required, but query results may be less accurate due both to false positives stemming from the descriptions of non existent data, and to false negatives due to the loss of relevant data descriptions whereas they are available in the system. If α is small, the update cost is high but there are few query results that refers to data no longer in the system, and nearly all available results are returned by the query.

Query cost We have seen that the use of summaries as data indexes may improve query processing. When a query Q is posed at a peer p, first it is matched against the global summary to determine the set of peers P_Q whose descriptions are considered as answers. Then, Q is directly propagated to those peers. As a consequence, the number of messages exchanged in the system is intended to be significantly reduced. Furthermore, the cooperation list associated with a global summary provides information about the relevance of each database description. Thus, it gives more flexibility in tuning the trade-off *recall* ρ / *precision* π of the query answers. Let V be the set of peers visited while processing a query. Then $\rho = |QS \cap V|/|QS|$ and $\pi = |QS \cap V|/|V|$, where QS is the set of all peers that really match the query (i.e. *Query Scope*).

The trade-off can be tuned by confronting the set P_Q with the cooperation list CL. The set of all partner peers P_H in CL can be divided into two subsets: $P_{old} = \{p \in P_H \mid p.v = 1\}$, the set of peers whose descriptions are considered old, and $P_{fresh} = \{p \in P_H \mid p.v = 0\}$ the set of peers whose descriptions are considered fresh according to their current data instances. Thus, if a query Q is propagated only to the set $V = P_Q \cap P_{fresh}$, then precision is maximum since all visited peers are certainly matching peers (no false positives), but recall depends on the fraction of false negatives in query results that could be returned by the set of excluded peers $P_Q \backslash P_{fresh}$. On the contrary, if the query Q is propagated to the extended set $V = P_Q \cup P_{old}$, recall value is maximum since all matching peers are visited (no false negatives), but precision depends on the fraction of false positives in query results that are returned by the set of peers P_{old}.

The above two situations are bounds of a range of strategies available to propagate the query. In our experiments, we assume $V = P_Q$, the initial peer set. Thus, the cost is computed as $C_Q = 2 \cdot |P_Q|$ number of messages.

5.2 Discussion

We evaluated the performance of PeerSum through simulation, using the SimJava package [7] and the BRITE [8] universal topology generator. We calibrated our simulator using real data gathered in [17].

In a first set of experiments we quantified the trade-off between query accuracy and the cost of updating a global summary. Interesting results showed that the fraction of stale answers in query results is limited to 3% for a network size lower than 2000 peers. For the update cost, we observed that the total number of messages increases with the number of peers, but not surprisingly, the number of messages per node remains almost the same. In the expression of the update cost C_{up}, the number of push messages for a given peer is independent of network size. On the other hand, the number of reconciliation messages decreases slowly with the number of peers, for a given value of the threshold α. More interestingly, when the threshold value decreases (from 0.8 to 0.3) we noticed a small cost increasing of 1.2 on average. However, a small value of the threshold α allows to significantly reduce the fraction of stale answers in query results. We concluded therefore that tuning our system parameter, i.e. the threshold α, do not incur additional traffic overhead in the system, while improving query accuracy.

In the second set of experiments, we compare our algorithm for query processing against non-index/flooding algorithms which are very used in real life, due to their simplicity and the lack of complex state information at each peer. Here, we limit the flooding by a value 3 of TTL (Time-To-Live). Our algorithm SI showed the best results that can be expected from any query processing algorithm, when no stale answers occur in query results (the ideal case). However, to give a real performance evaluation, we decided to study our algorithm in the worst case where the stale answers occur in query results. Even in that, SI showed a reduction of the number of messages, in comparison with flooding algorithms, that becomes more important with a large size of network. For instance, the query cost is reduced by a factor of 3 for a network of 2000 peers.

6 Conclusion

In this paper, we proposed PeerSum, a new service for managing data summaries in P2P and Grid systems. PeerSum supports scaling up in terms of two dimensions: number of participants and amount of data. As we discussed, our summaries are compact data descriptions that can approximately answer a query without retrieving original records from distributed databases. This is very interesting for Grid applications which tend to be more data intensive. On the other hand, as indexing structures, they support locating relevant data based on their content. Such semantic indexes are extremely efficient in large distributed systems, where accessing data becomes difficult and costly. Besides, we have addressed peer dynamicity which is critical in both P2P and Grid applications.

This paper made two main contributions. First, we defined a summary model for P2P systems, based on the SAINTETIQ process. SAINTETIQ generates database summaries with low complexity, and can be distributed and parallelized which makes it scalable in a distributed environment. Second, we proposed efficient algorithms for summary management in PeerSum. Our analysis and simulation results showed that

the use of summaries as data indexes reduces the cost of query routing by an important factor compared to flooding approaches, without incurring high costs in terms of update messages exchanged in the network. Furthermore, our system guarantees a good query accuracy which is measured in terms of the fraction of stale answers in query results. Moreover, tuning our system parameter, i.e. the freshness threshold α, improves query accuracy while inducing a small increasing of summary update cost.

References

1. K. Aberer *et* al. P-grid: a self-organizing structured P2P system. *SIGMOD Rec.*, 32(3), 2003.
2. R. Akbarinia, V. Martins, E. Pacitti, and P. Valduriez. Replication and query processing in the APPA data management system. In *Workshop on Distributed Data and Structures (WDAS'2004)*, 2004.
3. A. Chakravarti, G. Baumgartner, and M. Lauria. The organic grid: self-organizing computation on a peer-to-peer network. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 35(3):373–384, 2005.
4. A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In *Proc. of the 28 tn Conference on Distributed Computing Systems*, July 2002.
5. A. Crespo and H. Garcia-Molina. Semantic overlay networks for P2P systems. Technical report, Computer Science Department, Stanford University, 2002.
6. I. Foster and A. Iamnitchi. On death, taxes, and the convergence of peer-to-peer and grid computing. In *IPTPS*, pages 118–128, 2003.
7. F. Howell and R. McNab. Simjava: a discrete event simulation package for java with the applications in computer systems modeling. In *Int. Conf on Web-based Modelling and Simulation, San Diego CA, Society for Computer Simulation*, 1998.
8. http://www.cs.bu.edu/brite/.
9. http://www.ogsadai.org.uk. Open grid services architecture data access and integration.
10. http://www.snomed.org/snomedct.
11. G. Koloniari, Y. Petrakis, and E. Pitoura. Content–based overlay networks of xml peers based on multi-level bloom filters. In *Proc VLDB*, september 2003.
12. A. Oser, F. Naumann, W. Siberski, W. Nejdl, and U. Thaden. Semantic overlay clusters within super-peer networks. In *Proc of the International Workshop on Databases, Information Systems and Peer-to-Peer Computing in Conjunction with the VLDB 2003*, 2003.
13. T. Ozsu and P. Valduriez. *Principles of Distributed Database Systems*. Prentice Hall, 1999.
14. G. Raschia and N. Mouaddib. A fuzzy set-based approach to database summarization. *Fuzzy sets and systems 129(2)*, pages 137–162, 2002.
15. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content–addressable network. In *Proc SIGCOMM*, 2001.
16. R. Saint-Paul, G. Raschia, and N. Mouaddib. General purpose database summarization. In *Proc VLDB*, pages 733–744, 2005.
17. S. Saroiu, P. Gummadi, and S. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proc of Multimedia Computing and Networking (MMCN)*, 2002.
18. I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A scalabale peer-to-peer lookup service for internet applications. In *Proc ACM SIGCOMM*, 2001.
19. I. Tartinov *et* al. The piazza peer data management project. In *SIGMOD Record, 32(3)*, 2003.
20. A. Voglozin, G. Raschia, L. Ughetto, and N. Mouaddib. Querying the SAINTETIQ summaries-a first attempt. In *Int Conf. On Flexible Query Answering Systems (FQAS)*, 2004.
21. L. Zadeh. Concept of a linguistic variable and its application to approximate reasoning. *Information and Systems*, 1:119–249, 1975.